Seminario 2

Configuración e instalación de las herramientas de desarrollo

Introducción

A la hora de desarrollar una aplicación en el ámbito de la computación doméstica, lo normal es utilizar la misma plataforma en la que posteriormente se va a ejecutar, entendiendo por plataforma el conjunto de hardware más software de sistema. Por ejemplo, para desarrollar una aplicación que se ejecute en un procesador con arquitectura IA-32 y sistema operativo *Linux*, se instala la cadena de herramientas de desarrollo de GNU (GNU *toolchain*) en el mismo computador y se procede a su desarrollo.

Sin embargo, esta opción no es nada común en el desarrollo del software que se ejecutará en un sistema empotrado. Los sistemas empotrados se diseñan con unas características tan ajustadas a la función que van a desarrollar, que no tienen por qué tener la potencia o memoria necesarias para construir un programa, o puede que no dispongan de teclado o pantalla. Es más, seguramente no exista un compilador que se pueda ejecutar en el sistema empotrado. Por tanto, la opción más habitual para desarrollar el software empotrado consiste en usar una plataforma de desarrollo cruzado. El desarrollo se realiza en un computador anfitrión (*host*), normalmente un computador personal o una estación de trabajo, y el resultado final será un programa ejecutable en la plataforma de destino (*target*), el sistema empotrado.

Normalmente, la cadena de herramientas de desarrollo cruzado está formada por un compilador que genera ficheros ejecutables en la plataforma remota, una serie de utilidades para el manejo de estos ficheros binarios (ensamblador, enlazador, etc.), y por un depurador para comprobar que el código se ejecuta correctamente en el sistema empotrado. En algunos casos también será necesario el uso de un servidor de depuración, que hace posible la conexión entre el depurador (que se ejecuta en el PC) y el programa a depurar (que se ejecuta en el sistema empotrado). Se denomina *"cadena de herramientas"* (*toolchain*) porque normalmente las herramientas de desarrollo que la componen se usan en cadena. Estas herramientas se deben seleccionar según sea la arquitectura del procesador de nuestro sistema empotrado.

Objetivos

Los objetivos que se persiguen con el desarrollo de este seminario son:

- Entender la necesidad del desarrollo cruzado de aplicaciones para un sistema empotrado.
- Saber cómo instalar una cadena de herramientas de desarrollo cruzado.
- Familiarizarse con las herramientas de desarrollo.

Selección de la cadena de herramientas de desarrollo

El primer paso para seleccionar una cadena de herramientas de desarrollo cruzado consiste en identificar el procesador (o procesadores) para los que se va a desarrollar. Como a lo largo de las

prácticas vamos a usar como plataforma de destino la placa de desarrollo *Redwire Econotag*[1], este primer paso consiste en determinar el procesador en el que está basada la consola, el ARM7TDMI.

La mayoría de los fabricantes de procesadores tienen acuerdos con compañías que venden cadenas de desarrollo propietarias para sus procesadores. Sin embargo, si optamos por herramientas propietarias tenemos un par de inconvenientes. Por un lado está su alto coste, y por otro la dificultad que plantea el aprendizaje de unas nuevas herramientas de desarrollo.

Lo ideal sería usar siempre la misma cadena de herramientas de desarrollo para diferentes proyectos, de forma que aunque cambie el procesador empotrado, no se pierda tiempo en aprender a utilizar nuevas herramientas. Por tanto, puestos a escoger una cadena de herramientas de desarrollo, nos interesará que sea multi-plataforma. Además de esta característica, si andamos cortos de presupuesto, nos interesará que su precio sea bajo, y por último, también es deseable que la cadena de desarrollo genere código eficiente para la plataforma de destino, ya que normalmente, la memoria y la potencia del procesador de un sistema empotrado están muy ajustados.

La cadena de herramientas de desarrollo de GNU cumple todas estas características, ya que es gratuita, genera un código de gran calidad, y como se dispone de su código fuente, se puede construir para que genere código cruzado para multitud de plataformas, entre las que está la arquitectura ARM[2]. Por tanto, y teniendo en cuenta estas ventajas, usaremos esta cadena de herramientas para el desarrollo de las prácticas.

Componentes de la cadena de desarrollo

La cadena de herramientas de desarrollo que vamos a construir está compuesta por la colección de compiladores *gcc*, las utilidades de manipulación de ficheros binarios *binutils*, la biblioteca de funciones C *newlib*, el depurador *gdb*, y el servidor de depuración *OpenOCD*.

Los compiladores *gcc*

Realmente, *gcc* no es un único compilador, sino una colección de compiladores de diferentes lenguajes, como *C*, *C++*, *Java*, *Fortran*, *Ada*, etc., para diferentes plataformas (*IA-32*, *Alpha*, *ARM*, *MIPS*, *PowerPC*, *Sparc*, etc.)[3].

La mayoría de los programadores están acostumbrados al uso de algunos compiladores de esta colección (sobre todo los de C, C++ y Java) previamente construidos para generar código para la arquitectura del PC, ya que las distribuciones de *Linux* los incorporan como sus compiladores por defecto. Sin embargo, si en vez de instalar sólo los binarios de algunos de los compiladores de GNU, se instala el código fuente de *gcc*, se pueden construir compiladores de los lenguajes que queramos para multitud de plataformas, simplemente cambiando los parámetros de construcción de gcc.

Las *binutils*

Son una colección de utilidades para manejar ficheros binarios. Aunque las más importantes son `ld` y `as`, el enlazador y el ensamblador de GNU, hay otras bastante útiles:

- `addr2line` - Convierte las direcciones de los símbolos de un fichero objeto al nombre del

1 http://www.redwirellc.com
2 http://gcc.gnu.org/install/specific.html
3 *gcc* es el acrónimo de GNU *Compiler Collection*.

fichero fuente y número de línea

- `ar` - Utilidad para crear bibliotecas de funciones
- `gprof` - Muestra información útil sobre la ejecución de un programa para ayudar a su optimización
- `nm` - Genera una lista con los símbolos de un fichero objeto
- `objcopy` - Copia y cambia el formato de ficheros objeto
- `objdump` - Muestra la información de un fichero objeto
- `ranlib` - Genera un índice de los archivos contenidos en una biblioteca de funciones
- `readelf` - Muestra información de cualquier fichero en formato ELF
- `size` - Muestra el tamaño de las secciones de un fichero
- `strings` - Lista las cadenas imprimibles de un fichero
- `strip` - Descarta los símbolos de un fichero

El depurador *gdb*

Permite depurar el código que desarrollemos: consultar la memoria, inspeccionar el valor de las variables y registros, poner puntos de ruptura, etc. La única diferencia con respecto al uso que se hace tradicionalmente de *gdb* es que al hacer desarrollo cruzado, *gdb* se ejecutará en nuestro PC mientras que el programa que estemos depurando se ejecutará en el sistema empotrado. Por tanto, para facilitar la depuración será necesario conectar físicamente el sistema empotrado al PC (vía JTAG), y establecer un protocolo de depuración común (mediante *OpenOCD*).

La biblioteca de funciones C *newlib*

Es una implementación reducida de la biblioteca de funciones estándar de C, como las funciones de tratamiento de cadenas, de ficheros, de memoria, etc., junto con la biblioteca de funciones matemáticas, destinada a su uso en sistemas empotrados. Está mantenida por los desarrolladores de *RedHat* Jeff Johnston y Tom Fitzsimmons y se puede integrar dentro de gcc.

El servidor de depuración *OpenOCD*

Es un demonio capaz de comunicarse con la mayoría de procesadores de ARM que soporta múltiples tipos de interfaces JTAG de diferentes fabricantes. Está concebido para abstraer los detalles de la conexión entre el PC de desarrollo y el sistema empotrado y ofrecer a *gdb* un canal de comunicación estándar (TCP) y un protocolo de depuración único.

Instalación de las herramientas de desarrollo desde los repositorios

En el caso de que se esté utilizando un sistema operativo basado en *Linux*, se pueden obtener las herramientas de desarrollo directamente desde los repositorios. Para distribuciones basadas en *Debian/Ubuntu*, es necesario instalar los siguientes paquetes:

- `binutils-arm-none-eabi`
- `gcc-arm-none-eabi`

- `libnewlib-arm-none-eabi`
- `gdb-multiarch`[4]
- `openocd`

Configuración de la placa Econotag en Linux

Es necesario que añadamos una regla para el servicio *udev* de *Linux* que permita que los usuarios que no tienen privilegios de *root* puedan acceder a la *Econotag* a través del puerto USB. Para ello, debemos crear un fichero con el nombre `99-ftdi.rules` en el directorio `/etc/udev/rules.d` con el siguiente contenido:

```
# Regla para permitir a los usuarios acceder a la Econotag
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6010", MODE:="666"
```

Una vez creada la regla, debemos reiniciar el servicio *udev* para que haga uso de esta nueva regla. Para ello, ejecutaremos la orden `service udev restart`.

Prueba de las herramientas

Antes de seguir es conveniente comprobar que las herramientas se han instalado correctamente. Para ello usaremos la versión del *Hola Mundo* para la *Econotag*, cuyo código está accesible en la carpeta `/fenix/depar/atc/se/pracs/hello` desde los ordenadores del aula de prácticas. Para ello copiaremos el contenido de la carpeta `hello` a nuestro PC, por ejemplo a la carpeta `$HOME/se/pracs/hello`.

El *Makefile* de este ejemplo está preparado para usar la *toolchain* instalada desde los repositorios. Por tanto, asume que las herramientas están instaladas en el directorio `/usr` y que tienen el prefijo `arm-none-eabi`.

La construcción del programa se realiza con `make`. Si las herramientas están bien instaladas (y el *Makefile* ha sido modificado correctamente), se deberían generar los ficheros `hello.elf` y `hello.bin`, que son, respectivamente, el ELF y la imagen de nuestro *hola mundo* para la *Econotag*.

Para comprobar que *OpenOCD* se ha instalado correctamente, probaremos a subir la imagen del programa a la placa y ejecutarla. Para ello, conectaremos la placa al puerto USB del PC y escribiremos `make run`, lo que causará que se inicie el demonio de *OpenOCD*, al que se le pedirá que suba la imagen a la placa y la ejecute. Una vez terminado este proceso, deberíamos ver el led rojo de la placa parpadeando, lo que indicará que todo ha funcionado correctamente. Para detener la ejecución del programa podemos escribir `make halt` en la consola.

Depuración con gdb tui

Nos situamos en el directorio que contiene el ejecutable que queremos depurar (`hello.elf`), conectamos la placa al puerto USB y lanzamos el depurador, activándole la interfaz de depuración en modo texto:

```
gdb-multiarch -tui hello.elf
```
[5]

4 Si la distribución es antigua y no tiene este paquete, instalar `gdb-arm-none-eabi`.

5 Si hemos instalado `gdb-arm-none-eabi`, usaremos `gdb-arm-none-eabi` en lugar de `gdb-multiarch`.

Una vez dentro del depurador, indicamos que el código que va a depurar es de la arquitectura ARM:

```
set architecture arm
```

Conectamos el depurador a OpenOCD vía *pipe*:

```
target remote | openocd -c "gdb_port pipe" \
                        -f interface/ftdi/redbee-econotag.cfg      \
                        -f board/redbee.cfg
```

Hacemos un *soft reset* de la placa:

```
monitor soft_reset_halt
```

Desactivamos el controlador de interrupciones de la placa:

```
set *0x80020010 = 0
```

Subimos la imagen a la placa:

```
load
```

Ponemos un punto de ruptura en la etiqueta `_start`:

```
break _start
```

Y ejecutamos:

```
continue
```

Para agilizar el proceso, puesto que será igual en todas las prácticas, se puede usar un *script*. Por ejemplo, podemos colocar todas las órdenes de configuración del depurador en el fichero `init.gdb`:

```
set architecture arm
target remote | openocd -c "gdb_port pipe" \
                        -f interface/ftdi/redbee-econotag.cfg      \
                        -f board/redbee.cfg
monitor soft_reset_halt
set *0x80020010 = 0
load
break _start
continue
```

Y luego indicamos al depurador que cargue todas estas órdenes antes de cedernos el control:

```
gdb-multiarch --command=init.gdb -tui hello.elf
```

Una vez dentro de la interfaz del depurador, podemos cambiar las distintas ventanas, para mostrar el código fuente, los registros, el código ensamblador, etc., mediante la orden `layout`.

La lista de órdenes para el depurador se puede consultar en http://users.ece.utexas.edu/~adnan/gdb-refcard.pdf.

Para más información sobre la interfaz TIU de *gdb* consultar https://sourceware.org/gdb/onlinedocs/gdb/TUI.html.

en la *Econotag*.

Desarrollo de las prácticas en la ETSIIT

Las herramientas de desarrollo están instaladas en la versión de 32 bits del sistema operativo *Ubuntu 16.04* de las aulas de prácticas. Por otra parte, el código del programa *Hola Mundo* para la *Econotag* se encuentra en el directorio `/fenix/depar/atc/se/pracs/hello`.